

Abusing the Tutte Matrix: An Algebraic Instance Compression for the K-set-cycle Problem

Magnus Wahlström
Max-Planck-Institut für Informatik, Saarbrücken, Germany
wahl@mpi-inf.mpg.de

December 11, 2013

Abstract

We give an algebraic, determinant-based algorithm for the K -CYCLE problem, i.e., the problem of finding a cycle through a set of specified elements. Our approach gives a simple FPT algorithm for the problem, matching the $\mathcal{O}^*(2^{|K|})$ running time of the algorithm of Björklund et al. (SODA, 2012). Furthermore, our approach is open for treatment by classical algebraic tools (e.g., Gaussian elimination), and we show that it leads to a *polynomial compression* of the problem, i.e., a polynomial-time reduction of the K -CYCLE problem into an algebraic problem with coding size $\mathcal{O}(|K|^3)$. This is surprising, as several related problems (e.g., k -CYCLE and the DISJOINT PATHS problem) are known not to admit such a reduction unless the polynomial hierarchy collapses. Furthermore, despite the result, we are not aware of any *witness* for the K -CYCLE problem of size polynomial in $|K| + \log n$, which seems (for now) to separate the notions of polynomial compression and polynomial kernelization (as a polynomial kernelization for a problem in NP necessarily implies a small witness).

1 Introduction

Parameterized complexity [19, 21] is one of the major approaches for dealing with NP-hard problems. In this setting, the input is associated with a *parameter* k , usually (but not exclusively) either a parameter related to the solution size, or a structural parameter such as treewidth; the fundamental assumption is that problems with a smaller parameter value will be easier than general instances. The critical notion is that of an *FPT* algorithm, which runs in time $f(k) \cdot \text{poly}(n)$ for some $f(k)$ where $\text{poly}(n)$ is independent of k , i.e., the combinatorial explosion is confined to the parameter k . This notion has lead to a large number of interesting algorithmic principles; for some surveys, see, e.g., the Festschrift of Mike Fellows [7].

One of the most vibrant parts of parameterized complexity in recent years is the subfield of *kernelization*. A kernelization is one of the basic approaches for creating FPT algorithms: It is an algorithm which runs in time polynomial in both k and n , which reduces the size of the instance (e.g., via reduction rules such as “remove a vertex shown not to be required by the solution”) if the size is larger than some $f(k)$. Additionally, beyond being a design paradigm for FPT algorithms, it has been observed that the notion can be a good way to formalize effective *instance simplification*, e.g., preprocessing with a performance guarantee. A *polynomial kernel*, then, is a polynomial-time procedure which takes an input instance, with parameter k , and produces an output instance of size at most $\text{poly}(k)$, regardless of the value of n , without changing the problem status. Great interest has been taken in recent years in the question of which problems (and which problem parameterizations) admit polynomial

kernels. This was sparked by the creation of a lower bounds framework by Bodlaender et al. [8] and Fortnow and Santhanam [23]. These results provided a way to exclude the existence of a polynomial kernel, under the hypothesis that the polynomial hierarchy does not collapse. Later refinements and applications of this framework can be found in, e.g., [17, 10, 16, 25, 20, 18, 14]. Significant progress has also been made on the positive side; for a few examples, see [9, 40, 22]. A recent trend, relevant to the current paper, is the application of *algebraic* tools to kernelization, e.g., [29, 30]. (See related work, below.)

Sometimes, the results found by these investigations can be quite surprising. As an example, consider the problems VERTEX COVER (find a set of at most k vertices in a graph which covers all edges, i.e., a *vertex cover* of size at most k) and CONNECTED VERTEX COVER (find a vertex cover of size at most k which additionally is connected). The former is one of the most well-studied problems in theoretical computer science. In terms of parameterized complexity, it can be solved in time $\mathcal{O}^*(2^k)$ by a very simple algorithm, and in time $\mathcal{O}^*(1.28^k)$ by more involved means [12]. It has a simple 2-approximation, and a kernel of $2k$ vertices by the famous Nemhauser-Trotter theorem [36]. On the other hand, if the vertex cover is required to be connected, then the problem still has a simple greedy 2-approximation, an $\mathcal{O}^*(2^k)$ -time FPT algorithm [15], and, as shown by Dom et al. [18], no polynomial kernel unless the polynomial hierarchy collapses.

As another example, consider the following three problems. Given a graph G , find (a) a cycle with at least k vertices (the k -CYCLE problem); (b) a cycle passing through every element of a given set K , $|K| = k$ (the K -CYCLE problem); or (c) a cycle passing through every element of K , which furthermore passes the elements in a specified order (which we may dub the ORDERED K -CYCLE problem). Which of these seem more or less general? Which, if any, seems most likely to admit efficient instance simplification?

Let us make a quick review of known FPT and kernelization results for these problems. All are NP-hard; in the first two problems, setting $k = n$ yields the HAMILTONIAN CYCLE problem. The k -CYCLE problem is closely related to k -PATH (the problem of finding a path of length at least k), and there is by now a variety of interesting techniques that can be used to solve it in $2^{\mathcal{O}(k)} \text{poly}(n)$ time, from the seminal color-coding technique of Alon et al. [2], via the multilinear detection of Koutis [28] (see also Williams [43]), to the recent $\mathcal{O}^*(1.66^n)$ -time HAMILTONIAN CYCLE algorithm of Björklund [3], which was adapted to a parameterized setting in [5]. ORDERED K -CYCLE is equivalent to the well-known problem DISJOINT PATHS, where the input is k pairs of vertices (s_i, t_i) , and the question is if we can connect all pairs with pairwise vertex-disjoint paths. This problem seems much more challenging. Robertson and Seymour, in the context of the graph minors programme, showed that it is FPT; more specifically, that it can be solved in time $\mathcal{O}(n^3)$ for every fixed k [37]. Kawarabayashi et al. improved this to $\mathcal{O}(n^2)$ for every fixed k [27]. However, the algorithms are in both cases very involved, and the dependency of the running time on k is hard to pin down exactly, but at the very least multiply exponential. As for polynomial kernelization, both are infeasible: k -PATH and k -CYCLE were among the first problems to which the lower bounds framework was applied, and DISJOINT PATHS was addressed in [11]. In both cases, the conclusion is that neither problem allows a polynomial kernelization (or even a polynomial-time compression into size $\text{poly}(k)$) unless the polynomial hierarchy collapses.

The K -CYCLE problem, in turn, may intuitively seem to be closer in nature to the latter problem than the former – e.g., it is a terminal connectivity problem, parameterized by the number of terminals, and there is no obvious relation between the parameter and the size of the solution. Indeed, the problem can be solved via applications of the DISJOINT PATHS algorithm, and Kawarabayashi solved the problem in time $2^{2^{k^{10}}} \text{poly}(n)$ using graph minors-type graph structural reasoning [26]. However, recently, Björklund et al. [6] solved K -CYCLE

using an approach much closer to those of the cited k -PATH algorithms: they define a large polynomial, which can be evaluated in $2^k \cdot \text{poly}(n)$ time, and which, when evaluated over a field of characteristic two, is non-zero if and only if the instance is positive. The result then follows from an application of the Schwartz-Zippel lemma. (In fact, they solved the more general variant of finding a *shortest* K -cycle.) For kernelization, the status of K -CYCLE is so far unknown, but there are several factors – the lack of a small witness, the status of the related problems given above, the apparent difficulty of the problem – which would suggest that the answer should be negative (i.e., that K -CYCLE should have no polynomial kernel). As the present paper shows, this conclusion may well be mistaken.

Our results. We give an alternative algebraic algorithm for the K -CYCLE problem, also with a running time of $\mathcal{O}^*(2^{\mathcal{O}(k)})$, by encoding the problem into a variant of the Tutte matrix. More concretely, given G and K we construct a matrix M_G over $\text{GF}(2^\ell)$, whose entries are polynomials, and show that G has a K -cycle if and only if the determinant polynomial of M_G contains a certain type of term. Further minor modifications of the matrix yield an algorithm with running time $\mathcal{O}^*(2^k)$, and a matrix structure such that careful application of partial random evaluation and Gaussian elimination can reduce M_G to a matrix A with total coding length $\mathcal{O}(k^3)$, such that it can be decided from the determinant polynomial of A whether G has a K -cycle. All in all, this yields a randomized polynomial compression of K -CYCLE into space $\mathcal{O}(k^3)$. The construction, and all proofs, are simple, and we need only basic arguments about determinants and cycle covers to complete them.

We note that our approach so far fails to provide a polynomial kernel, in the strict sense; the reason being that the output is an instance of a different problem (of deciding a particular property of $\det A$) which is not known to be in NP, while a kernelization requires that the output is an instance of the same problem. This is closely related to the issue of the *witness size* required for K -CYCLE; we are not aware of a witness for either K -CYCLE or for our artificial algebraic output problem, of size $\text{poly}(k + \log n)$. We consider these results quite surprising.

Related work. The Tutte matrix (see Section 2) is a skew-symmetric matrix of indeterminates, created from the adjacency matrix of a graph G , which is non-singular if and only if G has a perfect matching [41]. This can be used to determine the size of a maximum matching in randomized time $\mathcal{O}(n^\omega)$ [32, 34], where $\omega < 2.3727$ is the matrix multiplication exponent [42, 39, 13]. Mucha and Sankowski [34] showed how to find a maximum matching in the same time. Geelen [24] gave a deterministic polynomial-time procedure which finds a maximum rank evaluation of the Tutte matrix (which does not lead to a competitive deterministic matching algorithm, but may be of interest for the general question of removing randomness due to applications of Schwartz-Zippel). For non-algebraic algorithms for matching, Micali and Vazirani [33] gave an algorithm that finds a maximum matching in general graphs in time $\mathcal{O}(m\sqrt{n})$.

Algebraic FPT algorithms, beyond those cited for k -PATH above, have been used by, e.g., Lokshtanov and Nederlof [31] and Cygan et al. [14]. See also Nederlof’s PhD thesis [35]. More specifically, algorithms based around determinant computations have been used by Björklund [4, 3]. However, we argue that the approach of the present paper leads to significantly simpler algorithms and correctness proofs than before. Algebraically based *kernelizations*, in particular using tools of matroid theory, have been given in [29, 30]. Related to the present work, it is interesting to note that the result of [29] was a pure compression, albeit within NP (the problem ODD CYCLE TRANSVERSAL was encoded into matroid, represented by a matrix of total coding length $\text{poly}(k)$), while [30] gave graph-based reduction rules, significantly broadening the applicability of the tools. A similar improvement on the tools of the present work would be highly interesting.

Organization. We review some basic definitions in the next section, then Section 3 gives, in turn, a very simple $\mathcal{O}^*(4^k)$ -time for K -CYCLE; an improvement to an $\mathcal{O}^*(2^k)$ -time algorithm; and the Gaussian polynomial compression.

2 Preliminaries

Parameterized complexity. A *parameterized problem* is a language $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$; the second component of instances (x, k) is called the parameter (cf. [19]). A parameterized problem is *fixed-parameter tractable* (FPT) if there is an algorithm A and a computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that A decides $(x, k) \in \mathcal{Q}$ in time $f(k)|x|^{\mathcal{O}(1)}$. A *kernelization of \mathcal{Q}* is a polynomial-time computable mapping $K: \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}: (x, k) \mapsto (x', k')$ such that $(x, k) \in \mathcal{Q}$ if and only if $(x', k') \in \mathcal{Q}$ and with $|x'|, k' \leq h(k)$ where h is a computable function; h is called the *size of the kernel* and K is a *polynomial kernelization* if $h(k)$ is polynomially bounded. A *polynomial compression* is a polynomial kernelization relaxed so that the output may be an instance of a (fixed) different language than the input language. This has also been called *bikernel* [1] and *generalized kernelization* [8]).

The Tutte matrix. Let $G = (V, E)$ be a simple undirected graph with $V = \{v_1, \dots, v_n\}$. The *Tutte matrix* A_G is the $n \times n$ matrix of indeterminates such that

$$A_G(i, j) = \begin{cases} x_{ij} & \text{if } v_i v_j \in E \text{ and } i < j, \\ -x_{ji} & \text{if } v_i v_j \in E \text{ and } i > j, \\ 0 & \text{otherwise,} \end{cases}$$

where x_{ij} are distinct commuting variables. Tutte [41] showed that $\det A_G \neq 0$ (viewed as a polynomial) if and only if G has a perfect matching. Lovasz [32] showed the applications of this type of result to randomized algorithms.

Determinants and cycle covers. We recall a few basic facts. Let $D = (V, E)$ be a directed graph, which may contain loops. A *cycle cover* of D is a set $\mathcal{C} \subseteq E$ of arcs such that every vertex in D has in- and out-degree exactly one in \mathcal{C} . We allow loops to be present in the cycle cover. For an undirected simple graph G , which again may contain loops, an *oriented cycle cover* of G is a cycle cover of the bidirectional graph corresponding to G . (Note that this implies that loops and isolated edges are permitted in the cycle cover, corresponding to cycles of length one respectively two.)

Let A be an $n \times n$ matrix over a field of characteristic two. Then the determinant and the permanent of A coincide:

$$\det A = \text{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n A(i, \sigma(i)), \quad (1)$$

where S_n is the set of all permutations of $[n]$. Let D be a directed graph on vertex set $V = \{v_1, \dots, v_n\}$ such that $v_i v_j \in E(D)$ if and only if $A(i, j) \neq 0$ (where $v_i v_i$ denotes a loop on the vertex v_i). There is a well-known bijection between terms of the naïve summation (1) of the determinant and cycle covers of D , as follows.

Proposition 1. *Let $D = (V, E)$ and A be as above. For a permutation $\sigma \in S_n$, let $\mathcal{C}_\sigma = \{v_i v_{\sigma(i)} : i \in [n]\}$. If $\mathcal{C}_\sigma \subseteq E$, then \mathcal{C}_σ is a cycle cover of D ; furthermore, this describes a bijection between cycle covers of D and non-zero terms in the summation (1).*

Let C be a simple cycle in a cycle cover \mathcal{C}_σ . We call C *reversible* if the cycle has length at least three and for every edge $v_i v_j \in C$, we have $A(i, j) = A(j, i)$; further, we call \mathcal{C}_σ reversible if it contains at least one reversible cycle. A critical observation, both in previous and present work, is that reversible cycle covers cancel in (1).

Proposition 2. *If (1) is computed over a field of characteristic two, then the terms corresponding to reversible cycle covers cancel each other.*

Proof. Let \mathcal{C}_σ be a reversible cycle cover, and let C be the first reversible cycle of \mathcal{C}_σ , counted by vertex incidence (i.e., the cycles of \mathcal{C}_σ are sorted according to the number of the earliest incident vertex). Let $\mathcal{C}' = \mathcal{C}_\sigma'$ be the cycle cover resulting from reversing C . Then this operation creates a fix-point-free involution among the reversible cycle covers. Further, as the terms of (1) corresponding to σ and σ' are identical by definition, all terms of (1) corresponding to reversible cycle covers will cancel each other out. \square

Thus, when reasoning about the surviving terms of $\det A$, we only need to concern ourselves with non-reversible cycle covers. (In particular, if $A = A_G$ is the Tutte matrix of a graph G over a field of characteristic two, then every cycle of length more than two is reversible, and there are no cycles of length one; thus the non-reversible cycle covers are exactly the perfect matchings of G .)

Schwartz-Zippel. We also recall the Schwartz-Zippel lemma.

Lemma 1 (Schwartz-Zippel [38, 44]). *Let $P(x_1, \dots, x_n)$ be a multivariate polynomial of total degree at most d over a field \mathbb{F} , and assume that P is not identically zero. Pick r_1, \dots, r_n uniformly at random from \mathbb{F} . Then $\Pr(P(r_1, \dots, r_n) = 0) \leq d/|\mathbb{F}|$.*

We will use this mostly, though not exclusively, for the case that P is the determinant of a matrix over $\text{GF}(2^\ell)$.

Detecting monomials in a polynomial. Our final generic ingredient is an application of inclusion-exclusion to finding certain monomials in a polynomial over a field of characteristic two. For a polynomial P and a monomial m , we let $P(m)$ denote the coefficient of m in P . We need a way to extract from P only those monomials divided by a certain term.

Lemma 2. *Let $P(x_1, \dots, x_n)$ be a polynomial over a field of characteristic two, and $T \subseteq [n]$ a set of target indices. For a set $I \subseteq [n]$, define $P_{-I}(x_1, \dots, x_n) = P(y_1, \dots, y_n)$ where $y_i = 0$ for $i \in I$ and $y_i = x_i$ otherwise. Define*

$$Q(x_1, \dots, x_n) = \sum_{I \subseteq T} P_{-I}(x_1, \dots, x_n).$$

Then for any monomial m such that $t := \prod_{i \in T} x_i$ divides m we have $Q(m) = P(m)$, and for every other monomial we have $Q(m) = 0$.

Proof. Consider a monomial m with non-zero coefficient in P . Observe first that for every $I \subseteq [n]$, we have $P_{-I}(m) = P(m)$ if no variable x_i with $i \in I$ occurs in m , and $P_{-I}(m) = 0$ otherwise. Now, if t divides m , then out of the $2^{|T|}$ evaluations, the monomial m occurs in exactly one (namely, $I = \emptyset$). Thus, $Q(m) = P(m)$. If t does not divide m , let $J = \{i \in I : x_i \text{ does not divide } m\}$, and observe that $P_{-I}(m) = P(m)$ for every $I \subseteq J$. Since $J \neq \emptyset$, this is an even number of occurrences of the same monomial with the same coefficient, which implies that they sum to zero. Applying this argument individually to every monomial in P accounts for all occurrences of monomials in the sum defining Q ; the result follows. \square

We remark that we do not require P to be multilinear (although we do require T to be a set rather than a multiset).

3 An Algebraic FPT Algorithm

We now give our algorithm and compression for K -CYCLE. Let us first fix a definition.

Definition 1. *For a vertex $v \in V$, a v -cycle is a cycle that passes through v . For a set $T \subseteq V$, a T -cycle is a cycle that passes through all vertices of T . In both cases, the cycle may pass through further vertices, but this is not required.*

The problem is then formally defined as follows.

K -CYCLE

Input: A graph $G = (V, E)$; a set $K \subseteq V$ of terminal vertices

Parameter: $k := |K|$

Question: Is there a K -cycle in G ?

We will show two algebraic FPT algorithms for this problem, giving two ways of encoding it into the determinant of a matrix. We then show how this implies a polynomial compression via Gaussian elimination, into space $\mathcal{O}(k^3)$.

3.1 Graph preprocessing

We begin with a simple preprocessing of the graph (reducing the terminals to degree two).

Lemma 3. *Let (G, K) be an instance of K -CYCLE with $|K| > 1$. We can reduce (G, K) to an equivalent instance (G', K') , $|K'| = |K|$, where $d(v) = 2$ for every $v \in K'$, and where K' is an independent set with no common neighbours.*

Proof. We assume that K is an independent set in G (by subdividing edges within K , if necessary). Construct G' from G by replacing every terminal $v \in K$ by two non-adjacent copies v', v'' (with neighbourhoods identical to that of v). Create a new vertex v with $N(v) = \{v', v''\}$. The new terminal set K' consists of these new vertices v .

It is easy to show that this reduction maintains the solution status. On the one hand, for any K -cycle in G , we may replace each portion $u - v - w$ of the cycle, with $v \in K$ and hence $u, w \notin K$, by a path $u - v' - v - v'' - w$, hitting the new terminal v . On the other hand, any K' -cycle in G' must pass through both neighbours v', v'' of each terminal $v \in K'$, and these neighbours are distinct for all terminals. Thus if each segment $v' - v - v''$ of the K' -cycle in G' is contracted into v , we get a valid K -cycle in G . \square

The requirement that $|K| > 1$ comes from the consideration of whether a single edge uv should be considered a K -cycle with $K = \{u\}$ (but the case $|K| = 1$ is in either case easily solvable in polynomial time).

For the rest of the paper, for convenience, we will let $G = (V, E)$ be a graph, on vertex set $V = \{v_1, \dots, v_n\}$ and terminal set $K = \{v_1, \dots, v_k\}$, to which the above reduction has already been applied. We also assume $N(v_i) = \{v_{k+2i-1}, v_{k+2i}\}$ for $i \in [k]$.

3.2 Matrix construction

We now show the matrix which will encode the existence of a K -cycle. We begin with a more intuitive construction, that implies a running time of $\mathcal{O}^*(4^k)$, then modify it to arrive at the $\mathcal{O}^*(2^k)$ -time algorithm and polynomial compression.

Given a graph G , reduced as per the previous subsection, we define the matrix A_G as follows. We start from the Tutte matrix A_G of G (although, as the field is of characteristic two, we will not observe the signs), and adjust so that $A(i, i) = 1$ for $i > 3k$ (effectively adding self-loops to all vertices except $N[K]$). Finally, we orient the edges incident to v_1 to make v_1 -cycles non-reversible: let $A(1, k+1)$ and $A(k+2, 1)$ be unmodified, but set $A(1, k+2) = A(k+1, 1) = 0$. This can be done safely, as any K -cycle of G can be oriented in either direction.

Let M_G denote the resulting matrix. We can detect a K -cycle in G as follows.

Theorem 1. *Let $T = \{x_{i,k+2i-1}, x_{i,k+2i} : i \in [k]\}$ and $t = \prod_{x \in T} x$. Then G has a K -cycle if and only if $\det M_G$, viewed as a polynomial, contains a monomial m with non-zero coefficient such that t divides m .*

Proof. Recall the summation (1) and the notion of a reversible cycle from Section 2. We claim a one-to-one correspondence between non-zero monomials of $\det M_G$ and non-reversible cycle covers.

This follows from basic observations, but we prove it for completeness. Since (1) is already in sum-product form, every non-zero monomial of $\det M_G$ corresponds to a non-empty set of summands from (1). By Prop. 2, we get the same result if we restrict ourselves to those summands corresponding to non-reversible cycle covers. We show that two summands, corresponding to distinct non-reversible cycle covers $\mathcal{C}, \mathcal{C}'$, always produce distinct monomials: if \mathcal{C} and \mathcal{C}' use distinct sets of underlying undirected, non-loop edges, then the claim is clear, and the set of loop edges of a cycle cover is a function of the set of non-loop edges. In the remaining case, \mathcal{C}' must be attainable by a reorientation of \mathcal{C} . However, there are by construction only three types of non-reversible cycles: loops, isolated edges, and v_1 -cycles, where a v_1 -cycle cannot be reversed, and loops and isolated cycles are invariant under reversal. Thus \mathcal{C} and \mathcal{C}' must produce distinct monomials.

The result is now simple. First, if \mathcal{C} is a K -cycle in G , then it contributes all factors in t , and by padding \mathcal{C} using self-loops we produce a non-reversible cycle cover, which produces a non-zero monomial of $\det M_G$. On the other hand, if a non-zero monomial in $\det M_G$ contains the factor $x_{i,k+2i-1}x_{i,k+2i}$, then in the corresponding cycle cover, the v_1 -cycle must also pass through v_i , as such a factor cannot be contributed by loops and isolated edges. By induction, if a non-zero monomial in $\det M_G$ is divided by t , then the corresponding cycle cover contains a v_1 -cycle which passes through every vertex of K , i.e., a K -cycle. \square

As $|T| = 2k$, this implies an $\mathcal{O}(2^{2k})$ -time randomized algorithm for the problem, via Lemma 2 and by evaluating the resulting polynomial Q randomly over $\text{GF}(2^\ell)$ for $\ell = \Omega(\log n)$. We will improve this in two ways: by introducing a modification which will let us match the $\mathcal{O}^*(2^k)$ running time of [6], and by showing how to use Gaussian elimination and partial random evaluation to produce a polynomial compression.

3.3 A 2^k Algorithm

We now show a different way to determine the existence of a K -cycle from M_G . Let an *orientation* of M_G be the result of, for every $v_i \in K$, $i > 1$, either setting $A(k+2i-1, i) = A(i, k+2i) = 0$ or $A(k+2i, i) = A(i, k+2i-1) = 0$, i.e., orienting the edges incident to v_i either as $v'_i \rightarrow v_i \rightarrow v''_i$ or as $v'_i \leftarrow v_i \leftarrow v''_i$. We claim the following.

Theorem 2. *Let Q' be the sum of $\det M'_G$ over all 2^{k-1} orientations M'_G of M_G . Then G has a K -cycle if and only if Q' is not identically zero.*

Proof. Let M'_G be an arbitrary orientation of M_G . As in Theorem 1, monomials of $\det M'_G$ correspond to non-reversible cycle covers, but now, every cycle incident on some $v_i \in K$ counts as non-reversible (and again, attempting to reverse such a cycle produces a zero-term). On the other hand, if two orientations M'_G and M''_G contain non-reversible cycle covers \mathcal{C}' and \mathcal{C}'' such that \mathcal{C}'' can be obtained by reorienting \mathcal{C}' , then \mathcal{C}' and \mathcal{C}'' contribute identical monomials to the sum, and their contribution may cancel. Thus, let \mathcal{C}^* be an *unoriented* cycle cover, such that every cycle in \mathcal{C}^* is either a loop, an isolated edge, or a cycle incident on K , and such that K is covered entirely by the latter type of cycles. We will count the number of contributions of orientations of \mathcal{C}^* to the sum.

For this, simply observe that in a single cycle C of \mathcal{C}^* , as soon as the orientation of at least one vertex of C has been determined, the direction taken through every other vertex of C is fixed as a consequence. Thus, if \mathcal{C}^* contains a K -cycle C , then only one orientation M'_G is possible, as the vertex v_1 enforces a direction already in M_G . On the other hand, if \mathcal{C}^* contains at least two cycles incident on K , then all cycles *not* incident on v_1 may be oriented arbitrarily, making for an even number of orientations, each one of which contributes the same monomial to the sum.

Thus non-zero monomials of Q' correspond to K -cycles in G , as promised. \square

This construction brings our algorithm closer in spirit to the determinant sums of Björklund [4, 3], or the algebraic FPT algorithms of Cygan et al. [15]. However, as the next subsection shows, by bringing the algorithm back into the structure of deciding properties of the determinant polynomial of a single matrix, we get a randomized polynomial compression for K -CYCLE via Gaussian elimination.

3.4 Polynomial Compression

Now, we finally show how to use the above for a polynomial compression of the K -CYCLE problem.

We describe one final modification of the matrix M_G . For every $v_i \in K$, $i > 1$, we introduce a new variable a_i , and multiply $A(k+2i-1, i)$ and $A(i, k+2i)$ by a_i , and $A(k+2i, i)$ and $A(i, k+2i-1)$ by $1-a_i$. Observe that this implies that the algorithm of Theorem 2 can be executed by iteratively setting each a_i to either 1 or 0, and computing the determinant each time. Strictly speaking, each individual determinant computation would then seem to require a fresh dose of randomness, via the Schwartz-Zippel evaluation step, making the approach inappropriate for kernelization. We show that it is possible to perform this in the alternate direction, first randomly evaluating every variable x_e for $e \in E(G)$, then performing Gaussian elimination into a compressed output, and finally (at some future time) performing the 2^{k-1} assignments to the variables a_i and computing the resulting determinants.

Theorem 3. *The K -CYCLE problem has a randomized polynomial compression of size $\mathcal{O}(k^3)$.*

Proof. We get the result in two steps, first showing that we can randomly evaluate the variables \mathbf{x} while leaving \mathbf{a} as indeterminates, then applying Gaussian elimination to produce a smaller matrix with the same determinant (viewed as a polynomial in \mathbf{a}).

Let $P(\mathbf{x}, \mathbf{a})$ be the determinant polynomial of M_G . Define $Q(\mathbf{x})$ to be the sum over the 2^{k-1} instantiations of \mathbf{a} necessary to emulate the algorithm of Theorem 2; observe that $Q(\mathbf{x})$ is a polynomial of degree n , and that $Q(\mathbf{x})$ is identically zero if and only if G has no K -cycle. Thus, again by Schwartz-Zippel, we may instantiate \mathbf{x} randomly from $\text{GF}(2^\ell)$, and with probability

at least $n/2^\ell$ the resulting values are such that the 2^{k-1} -sized evaluation of $Q(\mathbf{x})$ would return non-zero. Picking $\ell = \Theta(\log n)$ is sufficient for this step to succeed with polynomial probability in n (and with $\ell = \Theta(\log n + k)$, we get a failure rate still polynomial in n , but exponentially small in k). Note that by standard observations we may assume $k \geq \log n$, as otherwise the $\mathcal{O}^*(2^k)$ -algorithm runs in polynomial time.

Now, observe that we only need to know the existence of the polynomial Q for the above correctness argument. Thus, by replacing \mathbf{x} randomly by values from $\text{GF}(2^\ell)$, we get a matrix M' with mostly concrete values, and indeterminates in the top-left $3k \times 3k$ corner, such that preserving $\det M'$ is sufficient (up to the failure probability in the previous step) for preserving the information of whether G has a K -cycle.

Next, recall that row and column operations preserve the determinant of a matrix exactly. We show that we can reduce M' to a blocks form

$$M' = \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix},$$

where C is a matrix without indeterminates. Thus we will have $\det M' = (\det A)(\det C)$ where $\det C$ is a constant.

This is easy. For sets $R, C \subseteq [n]$, let $M[R, C]$ denote the induced submatrix of M with rows R and columns C . Observe that the submatrix $M_G[[3k+1, n], [3k+1, n]]$ is non-singular, as the diagonal contributes the term 1 to the determinant and every other term will contain at least one indeterminate. Thus (up to the failure probability), $M'[[3k+1, n], [3k+1, n]]$ is non-singular, and can be reduced to diagonal form with a non-zero diagonal, without introducing any new indeterminate entries in M' . Now we can use further row and column operations to reduce $M'[[1, 3k], [3k+1, n]]$ and $M'[[3k+1, n], [1, 3k]]$ to all-zero matrices (thereby modifying the contents of $M'[[1, 3k], [1, 3k]]$, but not $M'[[3k+1, n], [3k+1, n]]$). This creates the desired blocks form, and every step preserves the determinant precisely and is performed without further failure probability or growth of the individual entries (since we are working over a finite field).

Finally, we consider the resulting contents of the matrix A . Initially, the entries of $M'(i, j)$ for $i, j \leq 3k$ are either constants, or expressions $a_i \cdot c + c'$ for some constants c, c' . Every further row or column operation that modifies these entries adds some concrete value c' to the entry, meaning that we can maintain these entries in the form $a_i \cdot c + c'$ where c, c' are concrete values from $\text{GF}(2^\ell)$; thus the coding length remains $\mathcal{O}(\ell)$ bits per entry. We then multiply one arbitrary row of A by $\det C$, which again only has the effect of modifying the values c, c' . This gives us a $3k \times 3k$ matrix A' , with entries encoded into $\mathcal{O}(\ell) = \mathcal{O}(k)$ bits, such that $\det A' = \det M'$, where M' is the matrix produced by randomly instantiating \mathbf{x} in M_G . \square

Finally, we remark that, unusually, the output problem is not trivially in NP (as it is a question about the outcome of an exponentially large computation). Thus in terms of parameterized complexity, we do not strictly speaking get a polynomial kernel, as we know of no way of getting back from the matrix A' above to an instance of K -CYCLE.

4 Conclusions

We have shown an alternate algebraic algorithm for the K -CYCLE problem, recasting the original problem into a question about the existence of certain terms in the determinant polynomial of a matrix with indeterminate entries. By careful application of partial evaluation and Gaussian elimination, we have shown that this leads to a polynomial compression of a K -CYCLE instance into space $\mathcal{O}(|K|^3)$. This partially answers the question of the kernelizability of K -CYCLE, in a perhaps surprising direction.

Although we are not able to produce a proper kernel, since we are not able to get back to an instance of the K -CYCLE problem, such kernel-like polynomial compressions have been previously considered in parameterized complexity [1], and in fact all existing frameworks for excluding polynomial kernelization (e.g., [23, 20]) also exclude polynomial compressions. Thus, for the sake of a smooth theory, we hope that the K -CYCLE problem can also be shown to have a polynomial kernel (e.g., a compression within NP).

Another interesting improvement would be a more direct kernel, e.g., based on reduction rules which make direct modifications to G and K . The tools required for finding such rules may well have further applications (perhaps analogously to the two previous works [29, 30]).

It would also be interesting to consider further related problems, perhaps starting with the problems of finding a *shortest* K -cycle, and a K -cycle with a prescribed parity, as these problems can also be solved by the approach in [6]. While it seems that our algorithm can be adapted for this setting, it is not clear to us at the moment whether this can be done in a way that allows for a polynomial compression.

Acknowledgements The author is grateful to Thore Husfeldt and Stefan Kratsch for rewarding discussions, and to an anonymous reviewer for suggesting improvements to the paper.

References

- [1] N. Alon, G. Gutin, E. Kim, S. Szeider, and A. Yeo. Solving MAX- r -SAT above a tight lower bound. *Algorithmica*, pages 1–18, 2010.
- [2] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.
- [3] A. Björklund. Determinant sums for undirected hamiltonicity. In *FOCS*, pages 173–182, 2010.
- [4] A. Björklund. Exact covers via determinants. In *STACS*, pages 95–106, 2010.
- [5] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, arXiv:1007.1161, 2010.
- [6] A. Björklund, T. Husfeldt, and N. Taslamani. Shortest cycle through specified elements. In *SODA*, pages 1747–1753, 2012.
- [7] H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, editors. *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*. Springer, 2012.
- [8] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.
- [9] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) kernelization. In *FOCS*, pages 629–638, 2009.
- [10] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *STACS*, pages 165–176, 2011.
- [11] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

- [12] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.
- [13] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.
- [14] M. Cygan, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström. Clique cover and graph separation: New incompressibility results. In *ICALP (1)*, pages 254–265, 2012.
- [15] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159, 2011.
- [16] H. Dell and D. Marx. Kernelization of packing problems. In *SODA*, pages 68–81, 2012.
- [17] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *STOC*, pages 251–260, 2010.
- [18] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *ICALP (1)*, pages 378–389, 2009.
- [19] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, November 1998.
- [20] A. Drucker. New limits to classical and quantum instance compression. In *FOCS*, pages 609–618, 2012.
- [21] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, March 2006.
- [22] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In M. Charikar, editor, *SODA*, pages 503–510. SIAM, 2010.
- [23] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.
- [24] J. F. Geelen. An algebraic matching algorithm. *Combinatorica*, 20(1):61–70, 2000.
- [25] D. Hermelin and X. Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *SODA*, pages 104–113, 2012.
- [26] K. Kawarabayashi. An improved algorithm for finding cycles through elements. In *IPCO*, pages 374–384, 2008.
- [27] K. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.
- [28] I. Koutis. Faster algebraic algorithms for path and packing problems. In *ICALP (1)*, pages 575–586, 2008.
- [29] S. Kratsch and M. Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In *SODA*, pages 94–103, 2012.
- [30] S. Kratsch and M. Wahlström. Representative sets and irrelevant vertices: new tools for kernelization. In *FOCS*, pages 450–459, 2012.
- [31] D. Lokshtanov and J. Nederlof. Saving space by algebraization. In *STOC*, pages 321–330, 2010.

- [32] L. Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.
- [33] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.
- [34] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *FOCS*, pages 248–255, 2004.
- [35] J. Nederlof. *Space and Time Efficient Structural Improvements of Dynamic Programming Algorithms*. PhD thesis, University of Bergen, Norway, 2011. Available at <http://folk.uib.no/jne061/PhDthesisJesper.pdf>.
- [36] G. Nemhauser and L. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.
- [37] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.
- [38] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
- [39] A. Stothers. *On the Complexity of Matrix Multiplication*. PhD thesis, University of Edinburgh, 2010.
- [40] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.
- [41] W. T. Tutte. The factorization of linear graphs. *J. London Math. Soc.*, s1-22(2):107–111, 1947.
- [42] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *STOC*, pages 887–898, 2012.
- [43] R. Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.
- [44] R. E. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation (EUROSAM)*, pages 216–226, 1979.